Overcoming Challenges for Achieving High in-situ Training Accuracy with Emerging Memories

Shanshi Huang, Xiaoyu Sun, Xiaochen Peng, Hongwu Jiang, and Shimeng Yu
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA
Email: shimeng.yu@ece.gatech.edu

***Abstract*—Embedded artificial intelligence (AI) prefers the adaptive learning capability when deployed in the field, thus in-situ training on-chip is required. Emerging non-volatile memories (eNVMs) are of great interests serving as analog synapses in deep neural network (DNN) on-chip acceleration due to its multilevel programmability. However, the asymmetry/nonlinearity in the conductance tuning remains a grand challenge for achieving high in-situ training accuracy. In addition, analog-to-digital converter (ADC) at the edge of the memory array introduces an additional challenge - quantization error for in-memory computing. In this work, we gain new insights and overcome these challenges through an algorithm-hardware co-optimization. We incorporate these hardware non-ideal effects into the DNN propagation and weight update steps. We evaluate on a VGG-like network for CIFAR-10 dataset, and we show that the asymmetry of the conductance tuning is no longer a limiting factor of in-situ training accuracy if exploiting adaptive "momentum" in the weight update rule. Even considering ADC quantization error, in-situ training accuracy could approach software baseline. Our results show much relaxed requirements that enable a variety of eNVMs for DNN acceleration on the embedded AI platforms.**

Keywords—eNVMs, DNN, in-situ training, asymmetry and nonlinearity, ADC

I. Introduction

Embedded AI prefers the adaptive learning capability when deployed in the field. When new data is coming, the embedded AI platforms could fine tune its model without sharing the data back to the data center. This could potentially relieve the network bandwidth or privacy/security concerns. Therefore, implementing analog synapses on-chip that could be in-situ trained is an attractive solution. To this end, researchers are fascinated by the eNVMs that are capable of multilevel programmability. Today's DNN models tend to become deeper and more complicated with increasing number of parameters. As a result, the frequent data movements back and forth between the compute units and memory units becomes the critical bottleneck to the system performance and energy-efficiency. To alleviate this memory access bottleneck, in-memory computing paradigm is proposed to reduce data movements by emerging the computing units with the memory units [1]. Convolution operation contains vector-matrix multiplication (VMM), which takes up most of the computations in DNN. The crossbar nature of memory array supports analog VMM operations, which could boost the efficiency of convolution computation significantly with appropriate weight mapping to the memory cells by input sharing. In addition, in-memory computing could also improve the parallelism within the memory array by activating multiple rows, using the analog mechanism to conduct multiplication and perform current summation along bit lines (BLs).

Generally, both SRAM [2] and eNVMs [1] could be used for in-memory computing. Compared to the SRAM, eNVMs are more attractive for embedded AI platforms as they could keep stored parameters after power-off and do not need to transfer data to another storage device, and could be instantly turned on when receiving the awake signal. The eNVMs of interests include resistive random access memory (RRAM) [3], phase change memory (PCM) [4], ferroelectric field effect transistor (FeFET) [5], and electro-chemical random access memory (ECRAM) [6], etc. However, achieving high in-situ training accuracy using eNVMs for large-scale DNN remains a grand challenge today [7-8]. Material and device engineering solutions are still demanding. Alternatively, circuit techniques such as adding auxiliary transistors in the bit-cell (e.g. 3-tranisistor-1-capacitor [9] or 2-transistor-1-FeFET [10]) have been proposed, which may adversely increase area and power. Therefore, it is highly desirable to revisit the origins of this problem from the algorithm's perspective.

Fig. 1 shows the key challenges of in-memory computing with analog synapses. First, training with eNVMs suffers from non-idealities such as asymmetry/nonlinearity, device-to-device (D2D) variation and cycle-to-cycle (C2C) variation. The asymmetry is labeled as +/- for potentiation and depression (P/D) and the nonlinearity factor (NL) is labeled from 0 to 9. Definitions on these factors could be referred to our prior work [11]. These non-ideal effects make the conductance shift from the desired delta weight (ΔW) calculated by stochastic gradient decent (SGD) in backpropagation. Also, since the programming pulse to change the conductance could not be arbitrarily small, although the conductance of the devices are continuous, the states of the devices are limited. In addition, ADC at the edge of the crossbar array introduces quantization error for both forward and backward propagations. In this work, we perform a comprehensive study on these effects and seek hardware-aware algorithm solutions for in-situ training with eNVMs.

Capitalizing on the recent progresses in low-precision fixed-point training, we use WAGE [12] framework that quantizes the weight, activation, gradient and error as our algorithm baseline. WAGE is suitable for in-memory computing for two reasons. Firstly, it is a low precision training methods meaning that both the forward and backward propagations receive the quantized

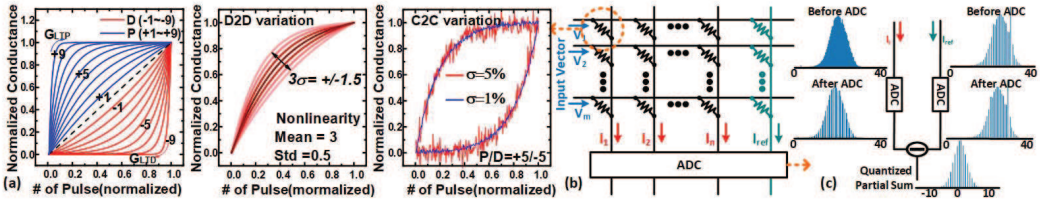

Fig. 1. Crossbar array for in-memory computing (b) with device non-idealities such as asymmetric and nonlinear conductance tuning, and device to device (D2D) variation and cycle-to-cycle (C2C) variation (a) and ADC quantization error (c). A reference column by subtraction is used to represent negative weights.

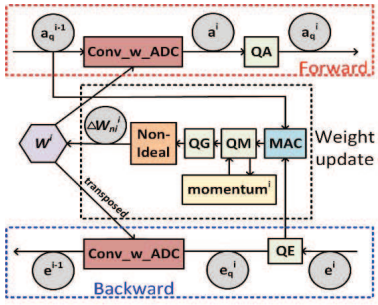

Fig. 2. Modified WAGE [12] training flow incorporating hardware non-ideal effects of layer i.

input for convolution. Secondly, it uses fixed-point quantization which limits the quantization range to [-1, 1] for the entire training process, while some other work [13] still uses floating range quantization which requires update of the quantization boundary for each iteration. Although the other work [13] could support ex-situ training for low precision inference, WAGE is more suitable for in-situ training on the device on the fly. In addition, WAGE uses stochastic quantization for gradient, which promises the convergence on low precision gradient. Our contribution to WAGE method is to incorporate the non-ideal effects of hardware into its training flow as shown in Fig. 2. The device non-idealities are added in the weight update stage and the ADC quantization is realized by modifying the convolution function as shown in the pseudo-code (Fig. 3).

The paper is structured as follows. In section II, we discuss the effects of each non-ideal effect and propose possible mitigation methods. Then, the simulation results are presented

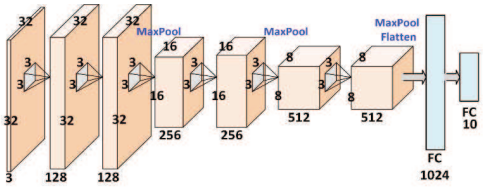

Fig. 4. VGG–like 8 layers neural network for CIFAR-10 dataset.

Modified WAGE Algorithm: Incoperting Non-ideal effects and ADC quantization.

Add Non-ideal Effect:

If quantize_momentum is True:

//Nomalize gradient calculated from the SGD to [-1,1]

$$g_n^i = normalize(g^i)$$

//Calculate the momentum based on the mometum factor

$$\Delta W^i(t) = \beta \Delta W_q^i(t-1) + (1-\beta)(-g_n^i)$$

//Quantize the momentum to reduce the overhead to save it.

$$\Delta W_q^i(t) = QM(\Delta W^i(t), M_p,)$$

//Stocahstic quantize to get pulse, the pulse width is deterimed by the gradient precision g_p, and the propability is scaled by the learning rate lr to control the percentage of weight update.

$$pulse^i = QG(\Delta W_q^i(t), g_p, lr)$$

If nonlinearity is True:

//map pulse to conductance change base on current value

$$\Delta W_{ni}^i = f(pulse^i, W^i, nl\_level)$$

If C2C variation is true

//W_{ni}^i is the real weight update applied to the weight corrsponding to the conductance change of eNVM cell.

$$\Delta W_{ni}^i = \Delta W_{ni}^i + \mathcal{N}(0, \sigma)$$

Add ADC quantization in Convolution:

//Binarize Input following two's complementary base

$$in = a^0 \cdot b^0 + a^1 \cdot b^1 + \cdots + a^n \cdot b^n$$

$$a^i \in [0,1],\ b^0 = -1, b^i = 2^{-i}\ \ 0 < i \le n$$

For i from 1 to n:

//Divide input and weight into k subarray according to the input channel depth

For j from 1 to k:

$$ps_{ij} = adc\left(conv(in_{ij}, w_j + 1)\right) - adc\left(conv(in_{ij}, 1)\right)$$

//sum partial sum across sub array

$$ps_i = ps_i + ps_{ij}$$

//sum partial sum across scale, by multiplied with scale b^i, the partial sum is converted back to decimal value

$$ps = ps + ps_i \cdot b^i$$

Fig. 3. Pseudo code for adding device non-idealities and ADC quantization into WAGE framework. Training is fully hardware-aware.

in section III to support our hypothesis. All of our evaluations are done for CIFAR-10 dataset with an 8-layer VGG-like network (Fig. 4).

II. Impact Factors for In-situ Training Accuracy

A. Asymmetritry/Nonlinearity

In the general DNN training process, weights are updated with the gradients calculated from SGD. For eNVMs-based accelerator, the gradients are mapped to the number of programming pulses to be applied on the device to change the weight (conductance). However, for most of realistic devices, the change is nonlinear respect to the pulse, and it will be based on the current conductance value. Generally the change will be large at the beginning when leaving maximum conductance Gmax or minimum conductance Gmin and will gradually saturate while approaching Gmin/Gmax, causing the conductance change nonlinear and asymmetric for increase and decrease. Recent work [14] shows that nonlinear but symmetric conductance tuning (up to P/D=+6/+6) just slightly hampers training accuracy, while nonlinear but asymmetric conductance tuning (with small P/D=+1/-1) causes significant accuracy loss (Fig. 5). While the prior works observed this phenomenon, no thorough analysis was done. Our hypothesis is as follows: when the device is approaching the Gmax by consecutive positive pulses, a negative pulse (as defined by a sign change in ΔW) will make a large drop of the conductance, as shown in Fig. 6 (a), as the asymmetric conductance tuning curve predicts. Similar argument could be applied when the device is approaching Gmin. In other words, it is statistically easier for the device to return the middle range of the conductance than approaching Gmax or Gmin. Fig. 6 (c) shows the ideal software weight distribution after the convergence, and Fig. 6 (d) shows the actual weight distribution (converted back from device conductance) when training with P/D=+3/-3. This result validated our hypothesis: weights are more uniformly distributed from -1 to +1 in ideal software training, while the weights are concentrated around 0 in actual device training. The weight distribution here is not like most of the floating-point training case which follows Gaussian distribution. We think this may be caused by the uniform initialization and the fixed boundary cut off manner of the WAGE quantization training method. Here we assume the weight +1/-1 are mapped to Gmax/Gmin, and a reference column is used for subtraction to represent the negative weights (Fig. 1 (c)).

With asymmetric/nonlinearity, if the global minima of the DNN's loss function landscape is at a location where some of the weights are near +1/-1, then any undesired sign change for ΔW will make it out of the global minima, since a sign flip means a large conductance change under asymmetry. There are three types of undesired ΔW sign change in the training process:1) sampling error of batches; 2) oscillation around local minima; 3) oscillation around global minima (Fig. 7).

To alleviate the problem of undesired sign change, we propose using adaptive "momentum" by increasing ΔW along the direction which has a constant ΔW sign. As shown in equation (1), ΔW applied at current step is the exponential moving average of the gradients from the history of previous steps plus the SGD value at current step. In this case, some

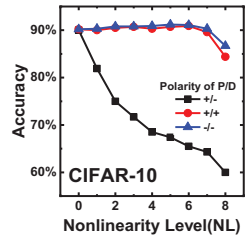

Fig. 5. Accuracy vs. NL level for symmetric and asymmetric weight update with SGD method.

interrupt sign flip could be avoided. β is defined as momentum factor, which decides the averaging window size.

$$\Delta W(t) = \beta \Delta W(t-1) + (1-\beta)\cdot(-\frac{\partial L}{\partial W}) \qquad (1)$$

By using momentum in weight update, the first two types of undesired sign change could be eliminated while oscillation around the global minimal is unavoidable during training. However, with the momentum and stochastic quantization, the probability of approaching global minima will be higher than that of leaving it. Fig. 6 (a) and (b) show the difference between the conductance change without and with momentum. Without momentum, the probability of updates totally depends on the current gradient from SGD independently. Any sign change of ΔW may lead to big jump-back of conductance (ΔG) towards the middle range of conductance. With momentum, for the example of weight increase, the probability of P update will be accumulated while approaching the target more smoothly. Even if for the current update step, the conductance change is not taken, the probability will accumulate as training goes on. The more steps on the same directions taken, the higher the probability to go toward that direction. When the conductance exceeds the target, the negative gradient by SGD will just

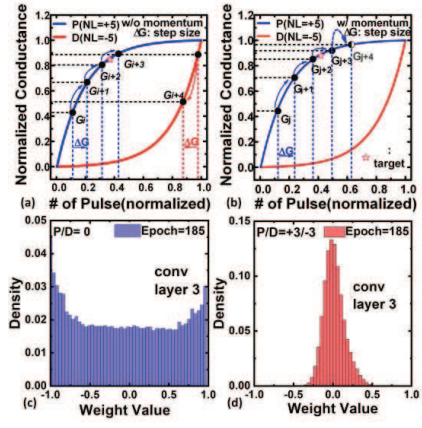

Fig. 6. (a) Conductance update trend without momentum. (b) Conductance update trend with momentum. (c) Weight distribution w/o nonlinearity. (d) Weight distribution with P/D = +3/-3 for conv layer 3 at epoch=185.

Fig. 7. Three types of undesired ΔW sign change during training.

decrease the P update probability, thus it may not cause jumping back while making the weight stay near the target. Although further increasing weight is also undesired, it will not go far away from the target because of the saturation nature of the P curve. As a result, the momentum compensate the asymmetry/nonlinearity of the conductance tuning if the global minima is located around Gmax/Gmin on some dimensions.

B. Update Step Size

Generally, eNVMs are analog synapses where cell conductance could be viewed as continuous variable between the dynamic range Gmin and Gmax, but the update step for the device could not be arbitrarily small as it is quantized by the programming pulse number. The applicable pulse number generally defines the weight precision (strictly speaking, the gradient precision since the weight is represented as conductance which could be located on any value because of the asymmetric nonlinearity and C2C variance of update). Table 1 surveys state-of-the-art device technologies ranging from 5-bit to 10-bit. Higher precision is preferred as smaller conductance update step size could reduce the impact of sign change under asymmetry. For the late stage of training, same update step is also necessary for exploiting the global minima. Using WAGE, we translate ΔW as the probability to apply a pulse through the stochastic quantization. Then the non-idealities of eNVMs are added to get the final weight update value ΔW_{ni}. Besides momentum, a learning rate (lr) is also used to modulate this probability after normalization to tune the percentage of weights updated at one time. The stochastic quantization is better than the deterministic quantization since it makes the weight update possible for small gradient. This is important to prevent the small gradient being diminished.

C. D2D Variation and C2C Variation

D2D variation is caused by that NL values differ across devices, this will not be a problem as the DNN model could have self-adapt to such static variation. C2C variation is due to the conductance variation upon each pulse. This could be a more severe problem if the C2C variation overwhelms the direction that is defined by the SGD. With significant C2C variation, a positive update step may end up with conductance decrease, resulting in an opposite direction of momentum move. Since C2C variation's standard deviation is related to the range of cell conductance, smaller step size will have higher probability of wrong direction update, which means low precision gradient and high nonlinearity level will be more robust to the C2C variation. Typical C2C values for some representative devices are shown in Table 1.

TABLE I. SURVEY OF REPRESENTATIVE ANALOG SYNAPSES REPORTED IN LITERATURE WITH WEIGHT PRECISION, P/D, AND C2C VARIATIONS.

Analog synapse reported	Weight (grad) precision	C2C	P/D
HZO FeFET[5]	32=5bit	0.5%	1.75/1.46
2T-1FeFET[10]	64=6bit	0.5%	0.85/0.85
2PCM+3T1C[9]	64=6bit	1.5%	0.2/-0.2
EpiRAM[15]	64=6bit	2%	0.5/-0.5
TaOx/HfOx[3]	128=7bit	3.7%	0.04/-0.63
ECRAM[6]	1000=10bit	<0.5%	0.347/0.268

D. ADC Quantization

The output of the VMM is represented as the current value on BL which is an analog signal. Although, in theory, eNVMs-based accelerator could take analog signal as input for the next stage [16], digital inputs are preferred for more reliable calculation. Also for training, the intermediate activation need to be saved for later usage in backpropagation and gradient calculation, the digital activation will be more suitable than the analog one. As the unrolled weight matrix is often larger than the memory sub-array size, ADC is used to digitize the partial sums from sub-arrays for later accumulations and propagations. To make the positive conductance to represent weight value which could be negative, read-out currents obtained from the weight columns and the reference column will go through ADCs first and then subtract digitally (Fig. 1 (c)). It is noted that unlike the partial sum value which is always zero centered, the positively accumulated current may have different means for different DNN layers and wider dynamic range.

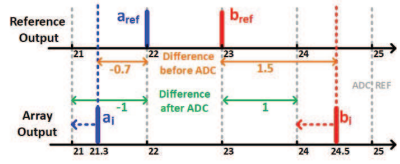

Fig. 8. ADC quantization example. Because of the round-down manner of ADC, their error will bias towards negative (e.g., -0.7 is quantized to -1), causing the output easier being cut by the subsequent ReLU function.

One drawback of ADC is that it has the round-down manner instead of the round-nearest operation that is used in most of the quantization algorithm, thus the partial sum tends to be biased towards negative (Fig. 8). Since we use ReLU as activation function, biasing towards negative will deactivate many neurons and prevent training from convergence. To solve this problem, we propose adding +1 to the LSB of all the negative partial sums, so that the positive results round down while the negative results round up. Such round-center quantization will have less errors than the round-down one as addition of partial sums from different sub-arrays will further accumulate the quantization errors. It is also possible to add +0.5 to mimic the round-nearest operation but it means higher precision required in hardware, so we do not use such method in our hardware-software co-simulation.

III. EVALUATION RESULTS

PyTorch platform is used for our simulation with the following default settings: 8-bit weight (gradient), 8-bit activation/error under the WAGE framework. For analysis with

nonlinearity/asymmetry, P/D=+3/-3 is used as default if without specification. 128×128 sub-array size are used for ADC quantization except the first layer. Batch size =200 for all cases except the big batch analysis. The software baseline accuracy is 92% for CIFAR-10 dataset without any non-idealities and partial sum quantization.

First, the effectiveness of momentum on compensating nonlinearity/asymmetry is studied. By sweeping momentum factor β, we found 0.9 is roughly the optimum choice, achieving 90% accuracy (Fig. 9(a)). The rest simulations are based on this value. On top of this momentum factor, in Fig. 9(b), we test the training performance with various NL factors under asymmetry. Applying the same learning rate will cause accuracy drop at high NL level since very big conductance change may happen when ΔW sign changes. By decreasing the learning rate, as a result of the stochastic quantization, the frequency of weight updates will decrease and reduce possible sign changes. The results show that reasonable accuracy (~87%) is still possible with the extreme P/D=+9/-9. To valid our hypothesis on the momentum's effect, we plot the weight distribution of the same layer with/without momentum at the same epoch. As shown in Fig. 10, with momentum, a more spread-out distribution is obtained than without momentum, resembling the ideal software training. Moreover, since the momentum for each weight should be stored (e.g. off-chip), it is also preferred to be quantized for saving the memory capacity. Fig. 9 (c) shows the quantization effect of momentum, and 10~12 momentum bits are required to avoid accuracy loss. This is higher than the gradient precision because for momentum we could not use stochastic quantization, and the momentum is to accumulate the small change with the same sign, it should have relatively high precision to avoid underflow.

We further test the effect of D2D variation assuming P/D=+3/-3 with 0.5 standard deviation (Fig. 9(d)) and found the

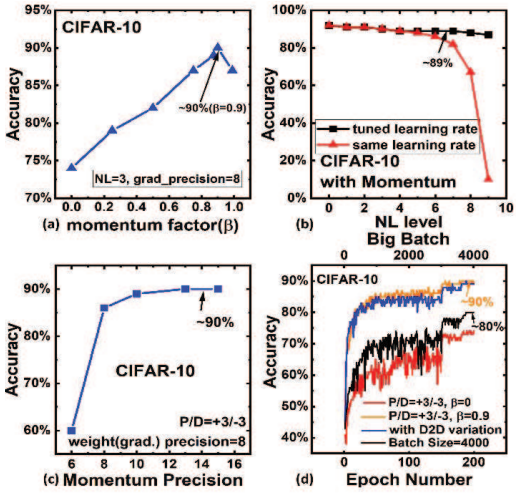

Fig. 9. (a) Accuracy vs. momentum factor β. (b) Accuracy vs. device nonlinearity factor under asymmetry. (c) Accuracy vs. momentum precision. (d) Training traces w/wo momentum, D2D variation and big batch size=4000.

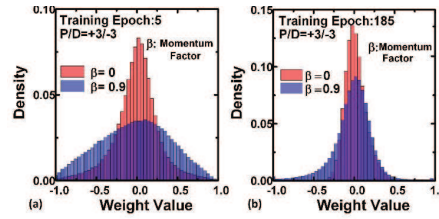

Fig. 10. Distribution of weights at initial training (epoch=5) and after convergence (epoch=185).

training trends is almost the same with the case of P/D=+3/-3 without variation. We also use very big batch size=4000 to train the network without momentum, which could reduce batch sampling error but could not solve the oscillation problem both around local and global minima. The black curve in Fig. 9(d) shows that big batch size improves the accuracy to some level compared to the small batch size (red curve) but could not achieve the same result as by momentum.

Fig. 11(a) shows that training with very low weight (gradient) precision down to 4-bit is feasible, if the learning rate could be fine-tuned. Fig. 11(b) shows for devices with high precision and low NL, high accuracy could be achieved without C2C variation, but accuracy drops rapidly with the increase of C2C variance from 1% to 5%. Instead, devices with low precision and high NL is more robust to C2C variation, though initial accuracy is lower. To understand this result, Fig. 12 shows one batch sampling of ΔWni vs. W based on the real conductance change. We could see that with smaller C2C variation, bigger NL or smaller weight (gradient) precision, there will be fewer points crossing over the positive ΔWni and the negative ΔWni boundary, which means less sign changes for the weight update.

For the ADC analysis in Fig. 13(a), by sweeping different ADC resolution bits, we found 8-bit will maintain a good accuracy ~87%. If we use round-down quantization instead of round-center, the ADC requirement increases to 11-bit. Compared to prior work [17] for inference only where 3-bit ADC is sufficient for a similar network, here we see a higher resolution is required for training. This is mainly due to two reasons. First, for training, the range of the partial sum may vary from iteration to iteration, so we just use linear quantization for easier implementation, which will require higher ADC precision than the nonlinear quantization based on the statistics. Second, in the simulation, weights are represented by only one analog

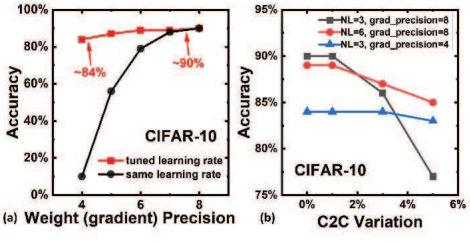

Fig. 11. (a) Accuracy vs. device weight (gradient) precision. (b) Accuracy vs. C2C variation. Momentum β =0.9 is applied.

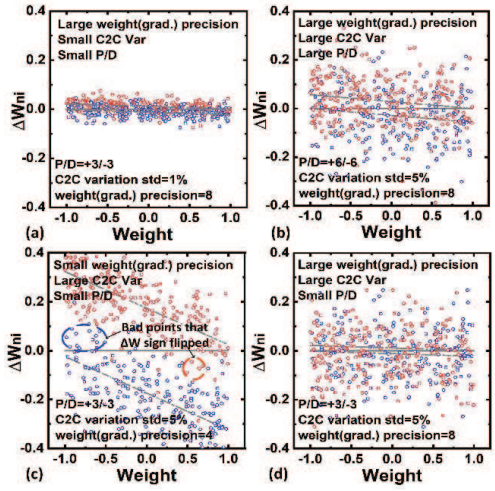

Fig. 12. Example of actual weight update pattern (ΔWni vs. W) in one batch for different P/D, C2C variation, and weight (gradient) precision. Due to C2C variation, the actual weight ΔWni update direction may be different than the momentum's sign. The red (blue) points are the actual conductance change for the cells that should have weight increase (decrease). Any red (blue) points falling into negative (positive) ΔWni means a wrong update direction. When C2C variation is large, better accuracy expected for (b) devices with large P/D and high precision or (c) devices with small P/D and low precision.

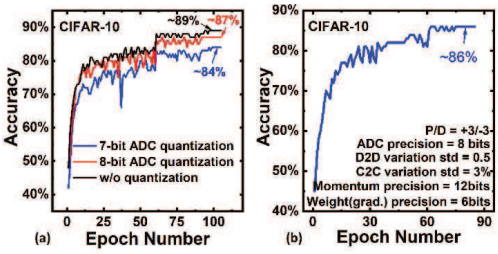

Fig. 13. (a) Training traces for different ADC resolutions. (b) Training traces when asymmetry/nonlinearity, D2D, C2C, momentum and ADC quantization effects all combined together.

synapse without any quantization which means the cell is already very high precision. Even if the ADC precision looks high, it has a big reduction from the full precision for the partial sum.

Lastly, we run the simulation with all the non-ideal effects combined, momentum quantization and ADC quantization. 86% accuracy is still achievable as show in Fig.13 (b), which is a remarkable result for in-situ training with practical eNVMs.

IV. Conclusion

In this work, we overcome the grand challenges for in-situ training with eNVM devices. First, asymmetry/nonlinearity is no longer a problem if momentum is introduced in the weight update. D2D variation is not a concern, either. Low-precision training is also possible. C2C variation is more problematic, device engineering to suppress C2C variation is needed. ADC resolution requirement is still high for on-chip training. This work could potentially pave a theoretical milestone for achieving high in-situ training accuracy with analog synapses.

Acknowledgement

This work is supported by NSF-CCF-1903951 and ASCENT, one of the SRC/DARPA JUMP centers.

References

[1] S. Yu, "Neuro-inspired computing with emerging non-volatile memory," *Proc. IEEE*, vol. 106, no. 2, pp. 260-285, 2018

[2] W.-S. Khwa, et al., "A 65nm 4Kb algorithm-dependent computing-in-memory SRAM unit-macro with 2.3 ns and 55.8 TOPS/W fully parallel product-sum operation for binary DNN edge processors," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018.

[3] W. Wu, et al., "A methodology to improve linearity of analog RRAM for neuromorphic computing," in *IEEE Symposium on VLSI Technology,* 2018.

[4] W. Kim, et al., "Confined PCM-based analog synaptic devices offering low resistance-drift and 1000 programmable states for deep learning," in in *IEEE Symposium on VLSI Technology,* 2018.

[5] M. Jerry, et al., "Ferroelectric FET analog synapse for acceleration of deep neural network training," in *IEEE International Electron Devices Meeting (IEDM),* 2017.

[6] J. Tang, et al., "ECRAM as scalable synaptic cell for high-speed, low-power neuromorphic computing," in *IEEE International Electron Devices Meeting (IEDM)*, 2018.

[7] G. W. Burr, et al., "Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power)," in *IEEE International Electron Devices Meeting (IEDM)*, 2015.

[8] S. Agarwal, et al., "Resistive memory device requirements for a neural algorithm accelerator," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2016.

[9] S. Ambrogio, et al., "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature,* vol. 558, pp. 60-67, 2018.

[10] X. Sun, et al., "Exploiting hybrid precision for training and inference: A 2T-1FeFET based analog synaptic weight cell," in *IEEE International Electron Devices Meeting (IEDM)*, 2018.

[11] P.-Y. Chen, et al., "NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures," in *IEEE International Electron Devices Meeting (IEDM)*, 2017.

[12] S. Wu, et al., "Training and inference with integers in deep neural networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[13] R. Banner, et al., "Scalable methods for 8-bit training of neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[14] X. Sun, et al., "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks," in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, 2019.

[15] S. Choi, et al., "SiGe epitaxial memory for neuromorphic computing with reproducible high performance based on engineered dislocations," *Nature Materials*, vol.17, pp. 335-340, 2018.

[16] H.-Y. Chang, et al., "AI hardware acceleration with analog memory: micro-architectures for low energy at high speed," *IBM Journal of Research and Development*, 2019.

[17] X. Sun, et al., "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *IEEE Design, Automation & Test in Europe (DATE) Conference*, 2018.